Microsoft DevOps Solutions: Designing and Implementing Health Checks

CLOUD APPLICATION HEALTH

David Tucker
TECHNICAL ARCHITECT & CTO CONSULTANT
@_davidtucker_ davidtucker.net

Objectives

Cloud Application Health

- Analyze system dependencies to determine which dependency should be included in health check
- Calculate healthy response timeouts based on SLO for the service
- Design approach for partial health situations
- Design approach for piecemeal recovery (e.g. to improve recovery time objective strategies)

Objectives

Monitoring Compute Health

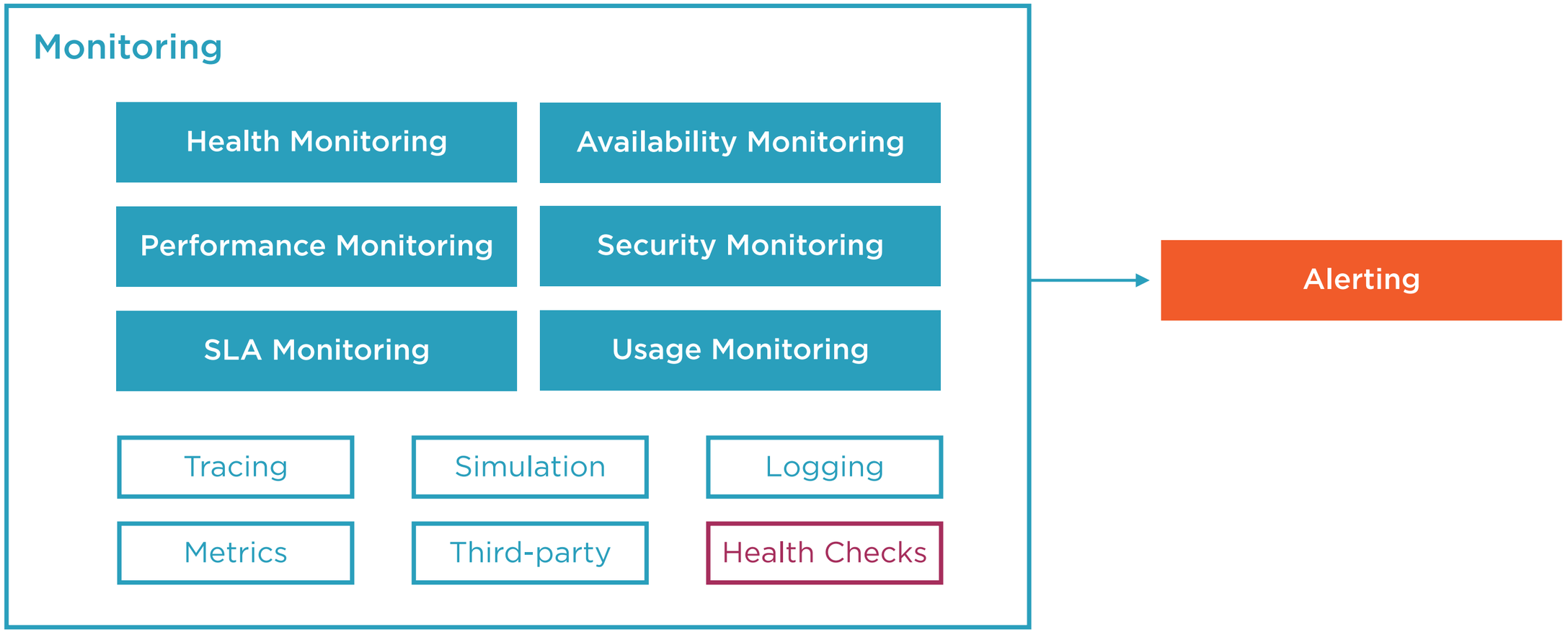
- Integrate health check with compute environment
- Implement different types of health checks (liveness, startup, shutdown)

Role of Monitoring and Health Checks

“A system is **healthy** if it is running and capable of processing requests.”

Microsoft Azure Documentation

Supporting Cloud Applications



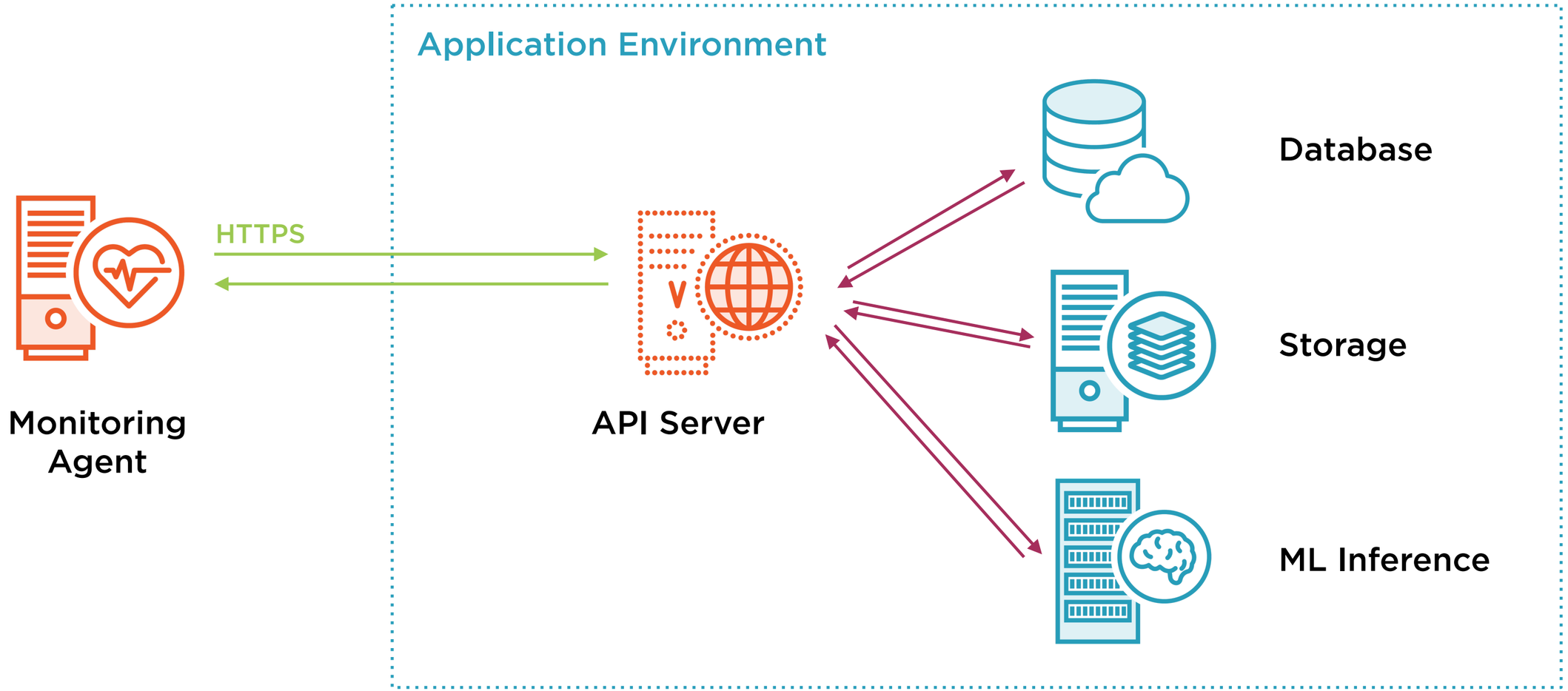
Application Health States

Green
All monitored systems are working as intended

Yellow
Partially healthy with some affected systems

Red
Application is non-functional or stopped

Anatomy of a Health Check



Health Check Elements

Health Check Endpoint
The checks performed by your application through a health check endpoint

Result Interpreter
The system calling and interpreting the health check response

Health Check Response

```
Response Time: 114 ms
200 OK

{
  totalDuration: 86,
  services: {
    storage: 45,
    database: 25,
    inference: 42
  },
  status: “green”
}
```

Application Health Points

Health Check Response

55 ms Timeout

Storage Service

55 ms 1.5 s

Database Service

45 ms 2.0 s

Machine Learning Inference Service

1.0 s 2.5 s

“A **service-level agreement** (SLA) is a commitment between a service provider and a client. Service-level agreements can contain numerous **service-performance metrics** with corresponding **service-level objectives**.”

Wikipedia

Service-level Objective Elements

Overall system availability - the percentage of time in which a service is available

Operational throughput - measure of the amount a system can handle

Operational response time - range of the time for a system to complete an operation

Application Health Points

Health Check Response

55 ms Timeout

Storage Service

55 ms 1.5 s

Database Service

45 ms 2.0 s

Machine Learning Inference Service

1.0 s 2.5 s

Tracking Application Dependencies

Azure Monitor

Analyze and diagnose issues across applications with Application Insights

Analyze and correlate infrastructure issues with Azure Monitor for VM's and Containers

Utilize Log Analytics for a deeper level of diagnostics

Provide both smart alerts and automated actions based on system condition

Enable visualizations with Azure dashboards and workbooks

Utilize Azure Monitor Metrics to collect data from system resources

"... if the overall system is depicted as **partially healthy**, the operator should be able to zoom in and determine which functionality is currently unavailable."

Microsoft Azure Documentation

Partial Health Questions

In situations where a health check indicates partial health:

- What systems are affected?
- What is the status of an affected system?
- What is the cause of the issue?

Types of Analysis

Hot Analysis
Time-critical analysis of current state and cause

Warm Analysis
Analysis of data leading up to health event

Cold Analysis
Analysis long-term data to determine overall trends

Dependency Tracking

Application Insights provides the ability to track and monitor calls to application-dependent systems. This can assist in determining what checks to include a health check as well as providing a tool to help with warm analysis of issues that arise.

Dependency Tracking

Application Insights SDK's for .NET and .NET Core provides automatic tracking

Supports tracking of the following dependencies:

- HTTP/HTTPS
- WCF (if HTTP bindings are used)
- SQL
- Azure Storage
- EventHub Client SDK
- ServiceBus Client SDK
- Azure Cosmos DB (if HTTP/HTTPS are used)

Manual Dependency Tracking

Some systems are not supported by default:

- Cosmos DB (if TCP is used)
- Redis

Dependencies can be tracked manually using the TrackDependency API

Application Insights Dependencies

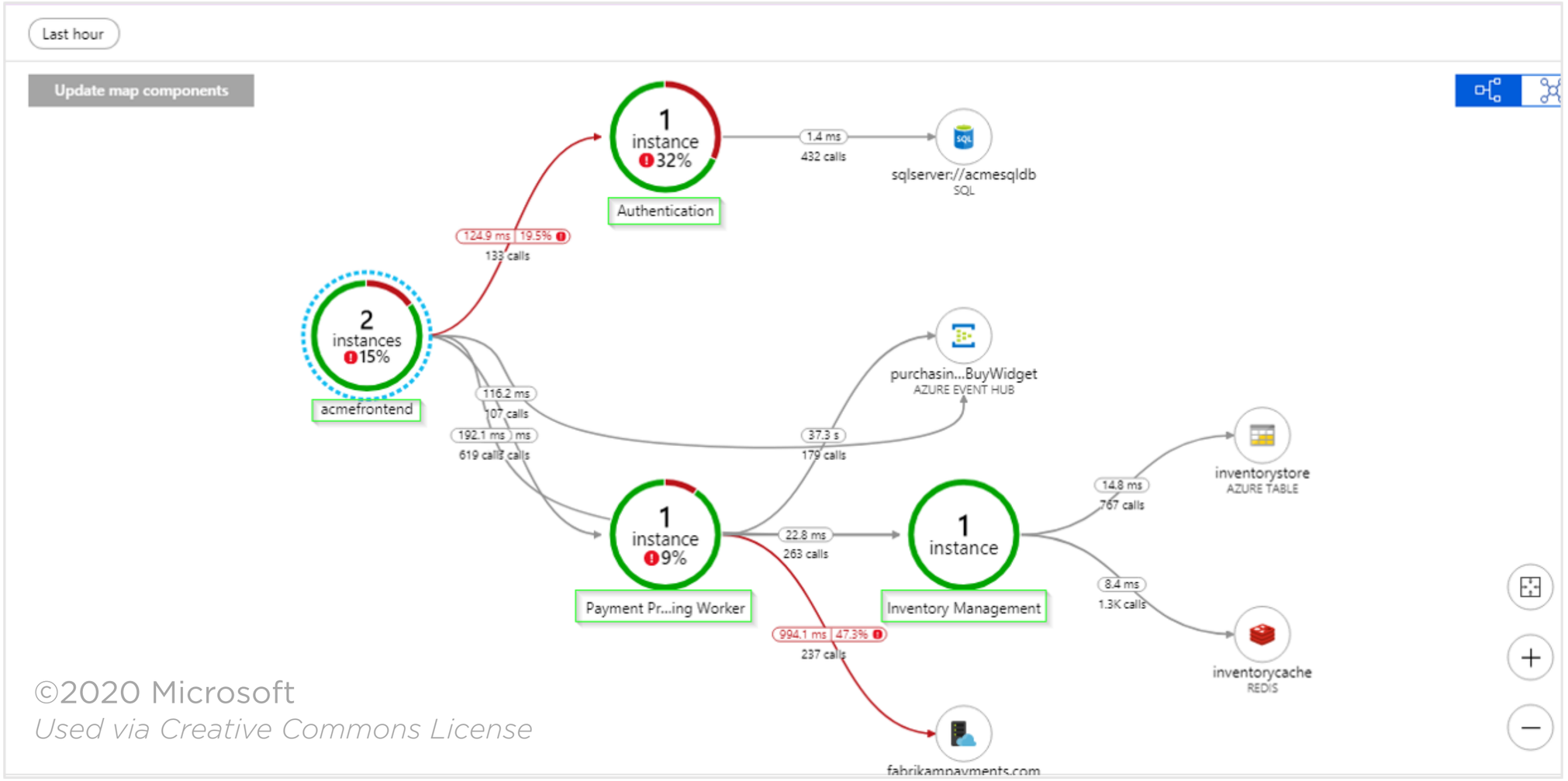
Application Map

Transaction Diagnostics

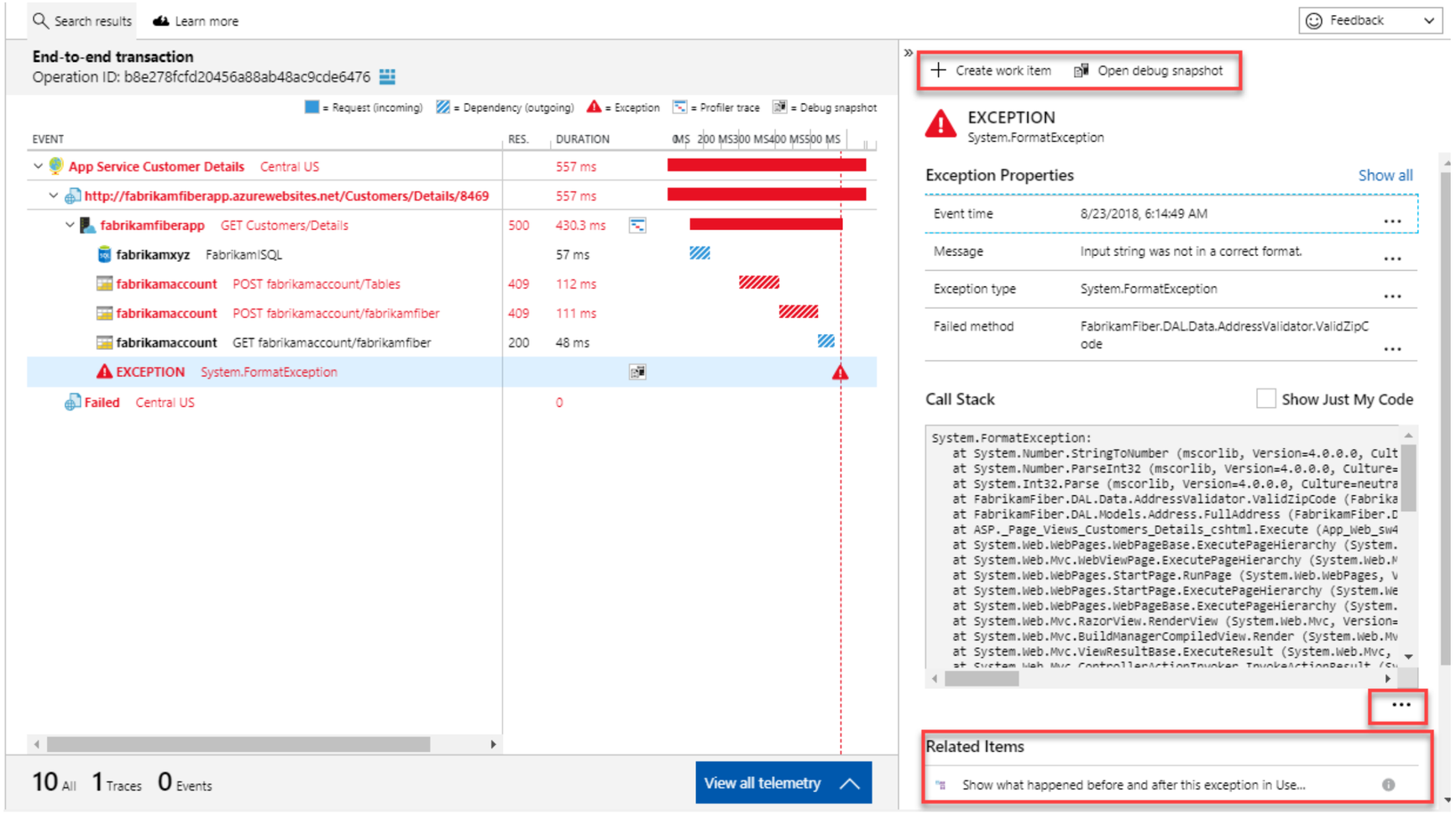
Browsers

Analytics

Application Insights Application Map



Application Insights Transaction Diagnostics



Recovering from Failure

“**Recovery time objective (RTO)** is the targeted duration of time and a service level within which a business process must be restored after a disaster (or disruption) in order to avoid unacceptable consequences associated with a break in business continuity”

Wikipedia

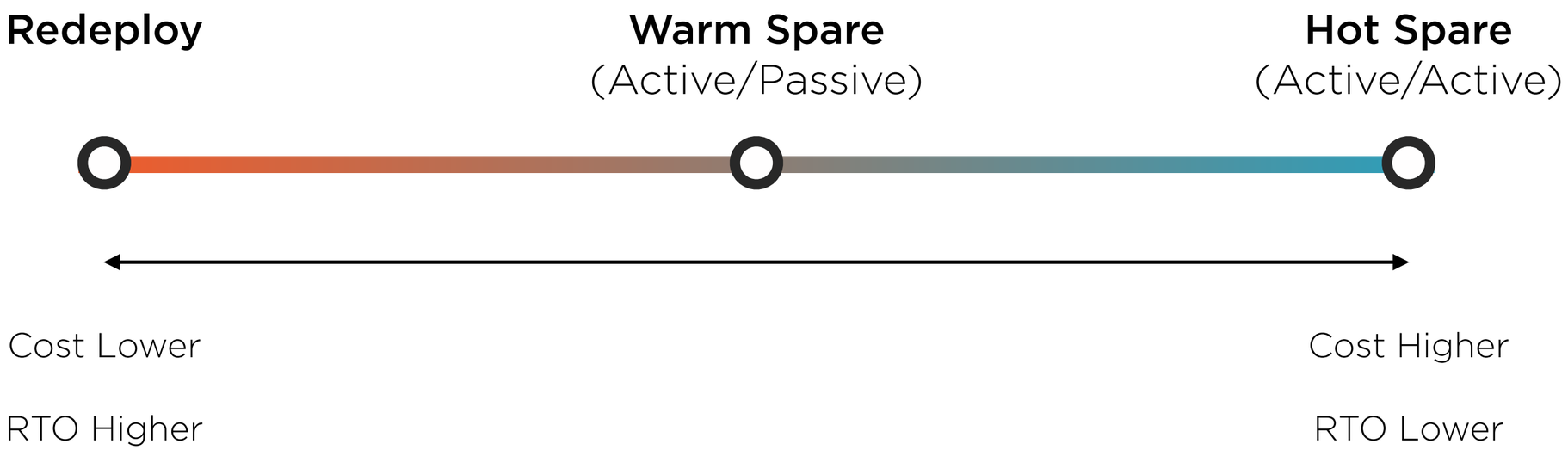
Recovery Considerations

Compute

Application Data

Managed Services

Application Recovery



Recovery Considerations

An effective data backup and recovery strategy is essential for a low RTO

Managed services with region replication reduce inner-region risk

Geo-replication for a service enables faster recovery when switching regions

Recovery must be practiced on a regular basis for there to be RTO confidence